So you're looking for a really small radio? How about building one onto a few chips? VE6VH walks us through the process and explains the theory behind it.

Design of a Software Defined FM Transceiver on an FPGA

(That's a Field Programmable Gate Array if You're an Analog Person)

BY MARTIN ALCOCK,* VE6VH

Software Defined Radio (SDR) is rapidly becoming the new norm in radio design, as it brings more flexibility, better performance, and a path to future upgrades as new modes become available, by simply adding a software upgrade.

Two forces have come together to enable this technology, the science of digital signal processing (DSP), and advances in processors that have enough raw horsepower to be able to implement DSP algorithms in real time. One of the candidate platforms for these algorithms is the *Field Programmable Gate Array* (FPGA), a modern-day approach to signal processing on an *Application Specific Integrated Circuit* (ASIC), which is well within the reach of the average amateur. These devices are a "blank canvas" to the digital designer, which can be utilized to realize computational components such as adders and multipliers that are required for signal processing at higher speeds than can be achieved in conventional processors.

Some of the newer FPGA devices have embedded "hard" processors, such as the popular core from *Advanced RISC Machines* (ARM), which can also be found in cell phones. RISC stands for *Reduced Instruction Set Computer*, which is a microprocessor that has a limited number of instructions so that it can operate at a higher speed (perform more millions of instructions per second). When an FPGA is coupled with such an embedded controller, it is called a *System on Chip*, or SOC for short.

These SOCs are of specific interest for the radio amateur as they offer the best of both worlds, and the manufacturer has added more functions to streamline transferring of data at high speeds between the two sides by making use of *Direct Memory Access* (DMA), which alleviates constant processor intervention.

In this article I will discuss the design of an FM transceiver on an SOC platform as a simple example of how the technology can be used, which can be dubbed FDR, for *FPGA Defined Radio*.

The Basic Theory of Software Defined Radio

The basic function of a radio, whether software- or hardware-defined, is to modulate and demodulate a carrier. Understanding how software plays a part in this first requires an understanding of the modulation and demodulation processes.

Engineers define the general form of a carrier as:

$$A_t \times cos(\omega_{ct} + \varphi_t)$$

Where A is the amplitude, ω_c is the center (carrier) frequency, and ϕ is the phase. We can modify (modulate) anything that has a letter "t" subscript on it, which means that we can change its amplitude, frequency, or phase with time, which we know as three forms of modulation. Such a carrier can be generated in software, but the highest frequency is limited by the speed of the

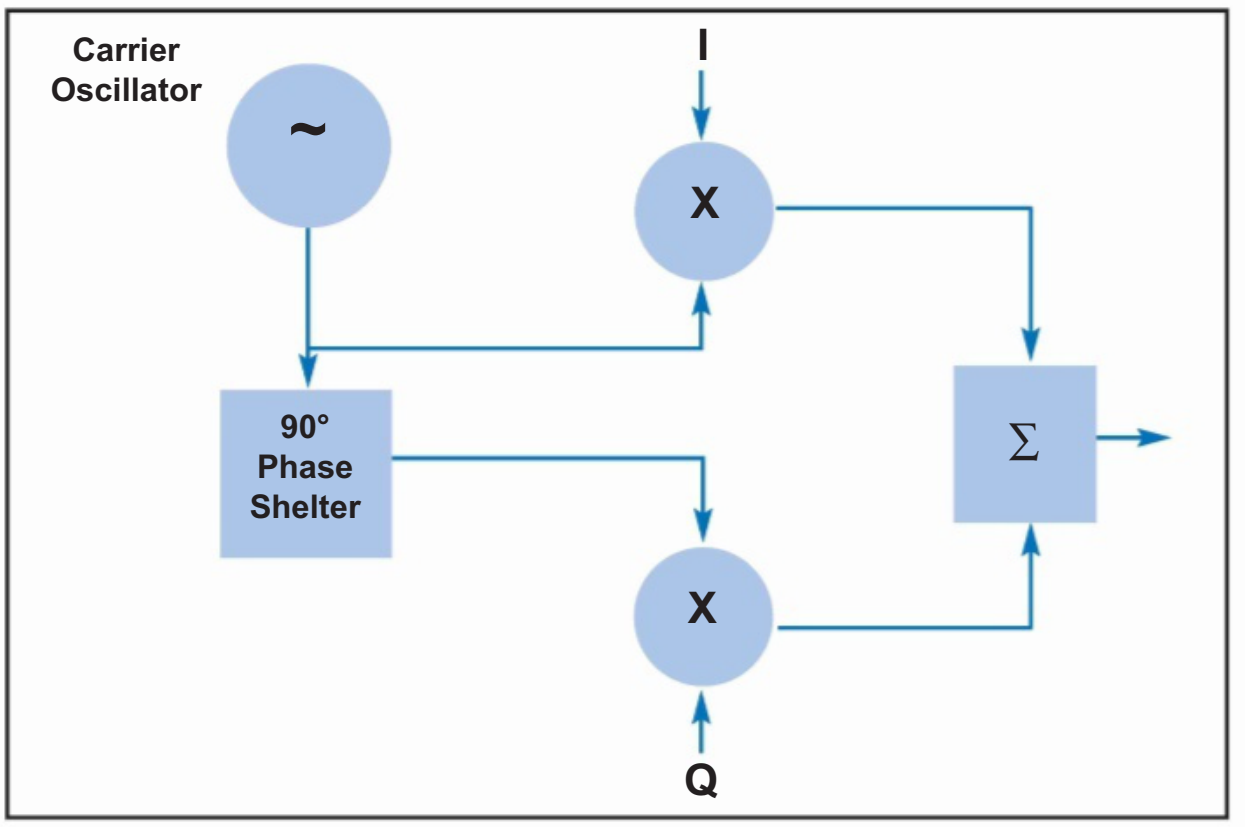


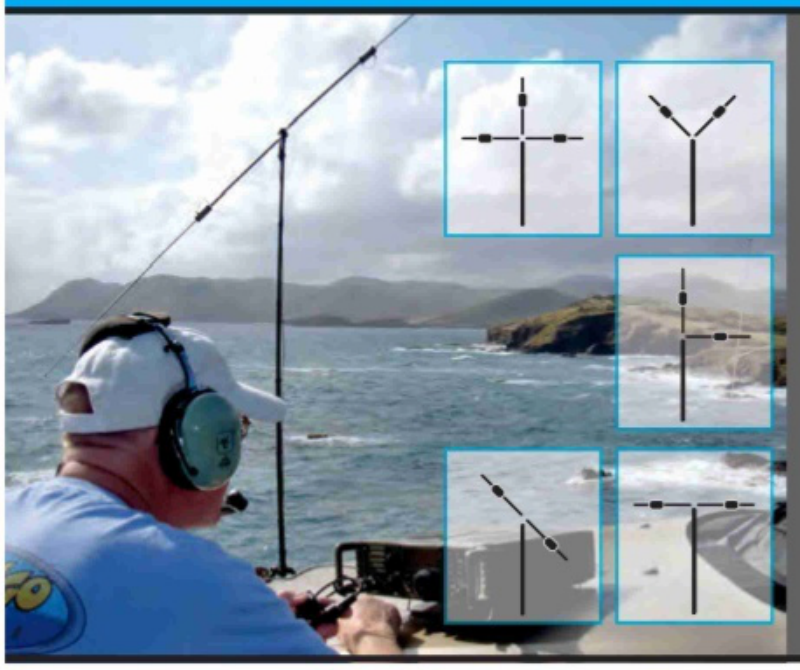
Figure 1. The universal modulator

* email: <ve6vh@ve6vh.org>

BUDDIPOLE

Secure online ordering at:
www.buddipole.com

HOT New Product
POWERMINI™ Back in Stock!



BUDDIPOLE FEATURES

› Multi-band design works 9 bands (40 meters thru 2 meters) with one set of adjustable coils!
› Modular Design – create dozens of different antennas with interchangeable parts
› Rotatable/Directional
› Lightweight, rugged components
› Rotating Arm Kit allows users to instantly change antenna configurations
› Used by Emergency Services Groups throughout the world
› Rated from QRP to 250 watts PEP

Compact Portable DC Power Management System with built-in Solar Controller

› High contrast OLED display
› Power management includes current and voltage readout
› User-definable Low Voltage Alarm + Cutoff
› RF Quiet Solar controller for use with panels up to 10 amps
› Rugged enclosure for field use
› Maximum current handling capacity – 25 amps

WHAT IS THE BUDDIPOLE?

The Buddipole™ Portable Dipole fits in your travel bag and assembles in minutes. The Buddipole is more than an antenna, it's a versatile system for launching your signal. Optimized for transmit power and proven for DX work, the Buddipole is the secret weapon used by HF portable operators all over the world.

3028 SE 59th Ct. #600
Hillsboro, OR 97123

tel: (503) 591 8001
fax: (503) 214 6802

info@buddipole.com

See our videos
www.youtube.com/buddipole

processor. To use this technique at VHF or UHF, a different method of modulating a carrier is required. An ideal solution would be to separate the modulation components from the carrier generation, create those in software and use hardware for the carrier generation. This can be easily realized by making use of a trigonometric identity, which rewrites the above equation as:

$$A_t cos(\varphi_t) \cos(\omega_{ct}) - A_t sin(\varphi_t) \sin(\omega_{ct})$$

Note that we have two carrier components and two similar modulation components. Both contain sine and cosine terms and are called the "In phase" and "Quadrature" components, or I and Q for short. The final modulation with the carrier can be carried out in hardware, using a "universal modulator," which is illustrated in *Figure 1*. It consists of an oscillator, which can be crystal controlled or a phase-locked loop, to generate the two carrier terms, which are independently modulated (multiplied) with the I and Q baseband components, and then summed together. This technique is known as "Zero IF" or "Direct to Baseband," as no intermediate conversion is utilized.

Conversely, the same technique can be utilized for demodulation in a slightly different configuration, as can be seen in *Figure 2*. The carrier is now mixed with the local oscillator but now in Quadrature. This is not unlike the familiar heterodyne techniques used in conventional radios, and the products of the mixing are the sum and difference of the inputs. A low-pass filter is therefore used to extract the desired I and Q signals and reject the others.

Implementing the Radio

The radio implementation will be made up of a baseband processor and an RF module. In the baseband processor, we will generate the required I and Q components for modulation and demodulate their counterparts in receive, while the RF module will generate and demodulate the carrier. The baseband processor chosen for this project was a DE10- Nano module from Terasic, Inc.[1], which is available at many different distributors and costs around $130. The FPGA is an Intel Cyclone V SOC, and the embedded *Hard Processor System* (HPS) is an ARM Cortex™-A9. The FPGA can be programmed using free tools from the manufacturer, and the HPS can be programmed as a "bare metal" device or run an operating system such as Linux, which can be booted from a Micro SD Card. A block diagram can be found in *Figure 3*.

For the RF module, there are several boards available off the shelf that feature single-chip Zero IF devices. They contain a phase-locked loop (PLL) for the quadrature oscillators, mixers, and low-pass filters; a low-noise amplifier (LNA) for receive and a small power amplifier (PA) for transmit, and output about +7dBm (5 milliwatts). I chose to use a Lime LM6002, which is available as a reference development kit called the "Myriad RF 1"[2] and sells for around $300. This board houses the RF chip and only requires an external reference oscillator for the PLL and has digital I/Q inputs and outputs and RF connections matched to 50 ohms. It can operate anywhere from 300 MHz to 3.8 GHz, which covers at least five of the UHF/SHF ham bands. A block diagram of that board is shown in *Figure 4*.

The missing piece of the puzzle was how to interconnect the RF and baseband components, but as there are two 40-pin uncommitted expansion headers on the DE-10, it was easy to link them using a mezzanine card, as shown in *Figure 5*.

This board contains mic and speaker amplifiers, an Analog Devices AD73311 front end, implements a power regulator, and provides connectivity to the Myriad board, an LCD display, a PTT input, and Squelch output. Completing the radio requires external components for RF switching in a simplex environment and a power amplifier, which can be built or purchased off the shelf. The board was realized with KiCAD, an open source *CAD* (computer-aided design) system that is maintained by CERN in Switzerland and freely available.

Baseband Processing

A block diagram of the baseband processing that was implemented in the FPGA is shown in *Figure 6*. Starting at the left hand side, the mic audio is first digitized by the analog front end, then makes its way through the modulator (ignore the interpolators and decimators for now), to the Myriad chip for modulation. Conversely, the receive I and Q components are filtered for selectivity, demodulated, and then converted back to analog signals at the front end. The Myriad board is separately controlled by the ARM microcontroller, which has an *SPI* (Serial Peripheral Interface) connection to program the PLLs and control other functions.

FM Modulator

From the universal modulator equations earlier, we know that we can easily modulate the phase of our carrier by generating the required I and Q components. However, we want to modulate the frequency, not the phase, but fortunately these are related. As we know that frequency is defined as the rate of change of phase (known to mathematicians as the "derivative"), therefore the inverse is also true, that phase is the integral of frequency. In other words, we can create our frequency offset with a continuous sinusoid in quadrature. For example, to achieve an 8-kHz deviation, we must generate an 8-kHz sinusoid; a 6-kHz sinusoid for 6-kHz deviation, and so on.

Figure 7 illustrates a simple quadrature oscillator using a lookup table that can be implemented on an FPGA. It consists of an amplitude-to-phase converter, followed by an integrator, which is an accumulator. The FPGA has on-chip memory that is dual-ported and has two read and write ports that operate independently. The memory contains a single sine wave, and we use an address-offset technique to index into the appropriate cosine and sine components.

This technique is known as *Direct Digital Synthesis* (DDS), where the output frequency can be calculated from the following formula:

$$Fo = \frac{n}{rom\ size} \times Fs$$

where F_o is the output frequency, F_s is the sample rate, and n represents the output frequency as a fraction of the ROM size. By setting n to the appropriate value for a specific analog amplitude, we can achieve frequency modulation.

Implementing the Demodulator

In the demodulator we want to recover the phase angle, and that can be simply done with a differentiator, as illustrated in *Figure 8*.

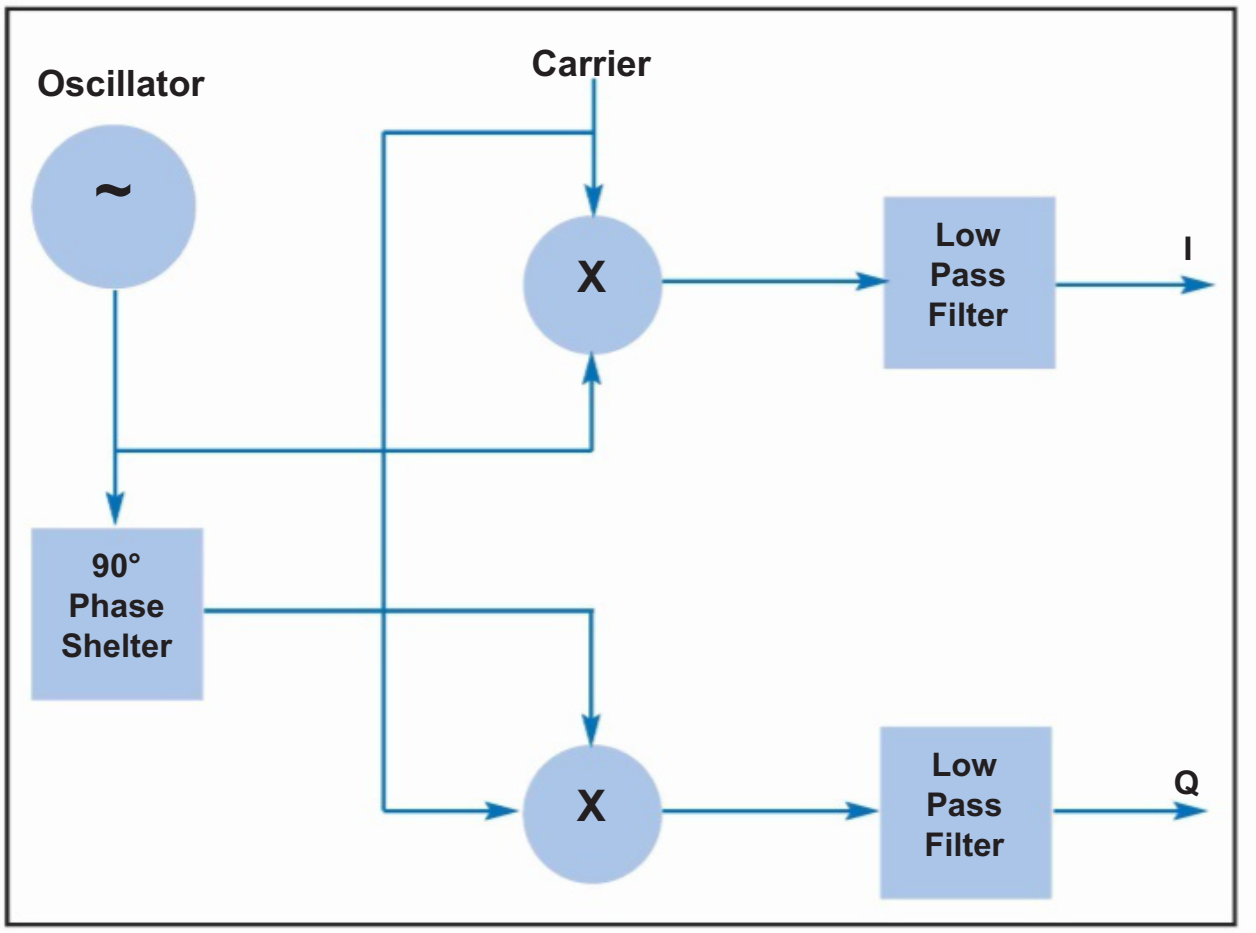


Figure 2. Universal demodulator

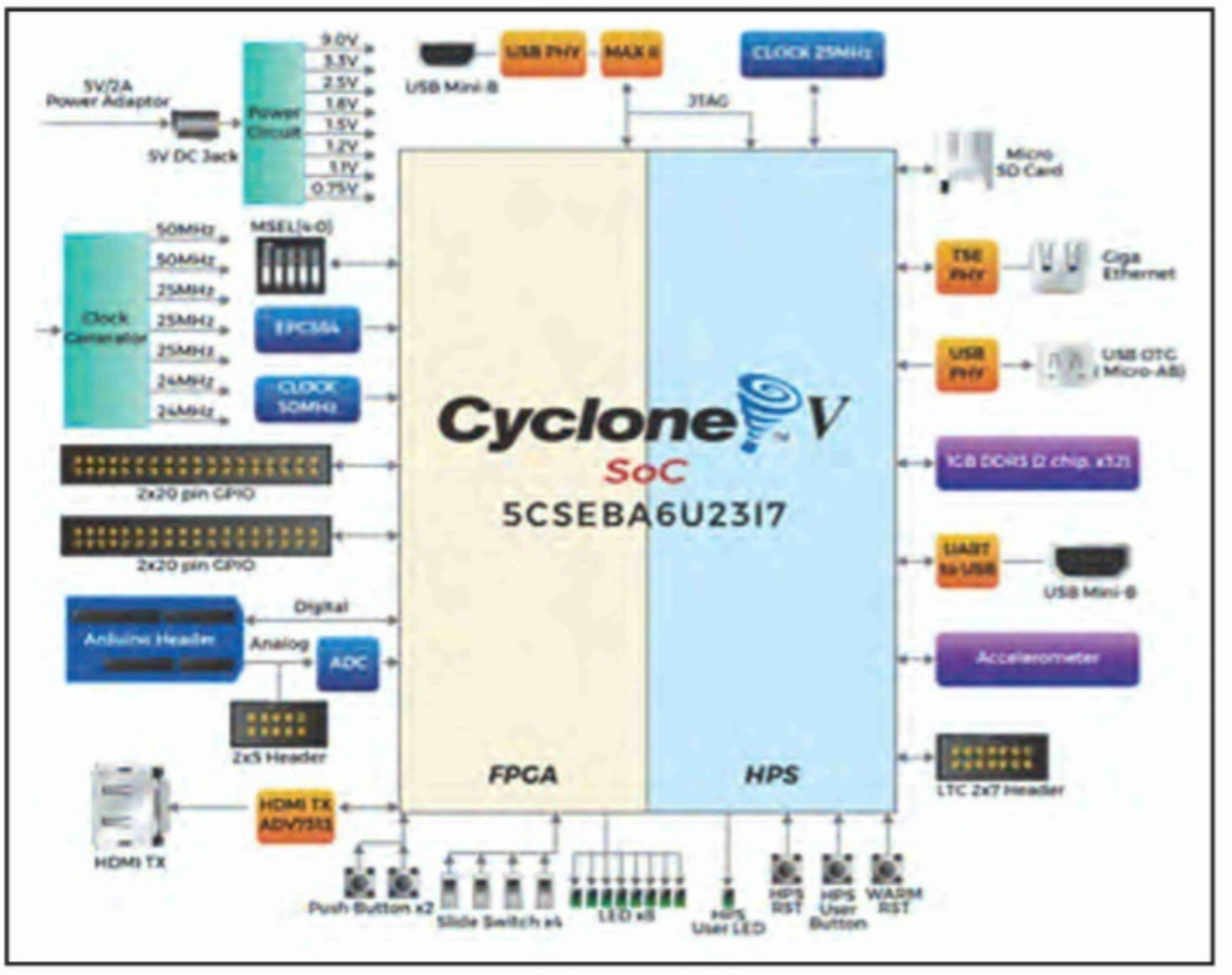


Figure 3. DE10-Nano FPGA platform (Courtesy Terasic, Inc. See Note 1)

To accomplish the differentiation, we need to use both the I and Q terms from the current sample and the previous one, which requires a delay. In signal processing terms, we use the symbol z to represent the sample delay operator, so z-1 represents a single sample delay, z-2 would represent 2, and so on. The equation for the demodulator would then be:

$$I \times Q^{z-1} + Q \times I^{z-1}$$

The result of this is the sine of the phase angle, and the amplitude can then be recovered by using an inverse sine (arcsine) lookup table.

Discrete Time, Sample Rates, Interpolation, and Decimation

In our radio design, we start by taking an analog signal from a microphone and digitizing it using our front-end device. The mic signal exists in what is called the "continuous" time domain, which means that you can attach it to an oscilloscope and observe the waveform. After sampling, it is now in what is called the "discrete" time domain, which has one sample for each time step, determined by the rate at which we sample the signal. A researcher named Harry Nyquist determined that to be able to properly sample a continuous signal and accurately reproduce it, we need to have at least two samples, so the min-

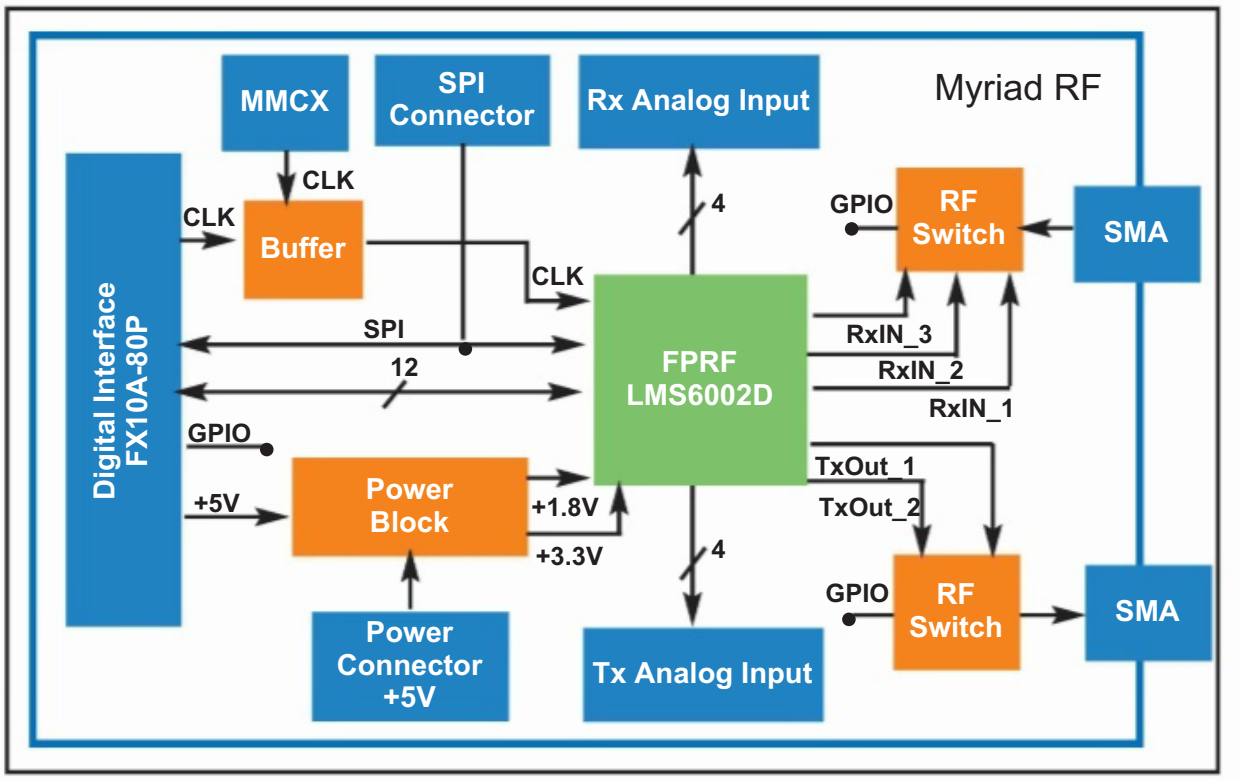


Figure 4. Myriad RF board block diagram (see Note 2)

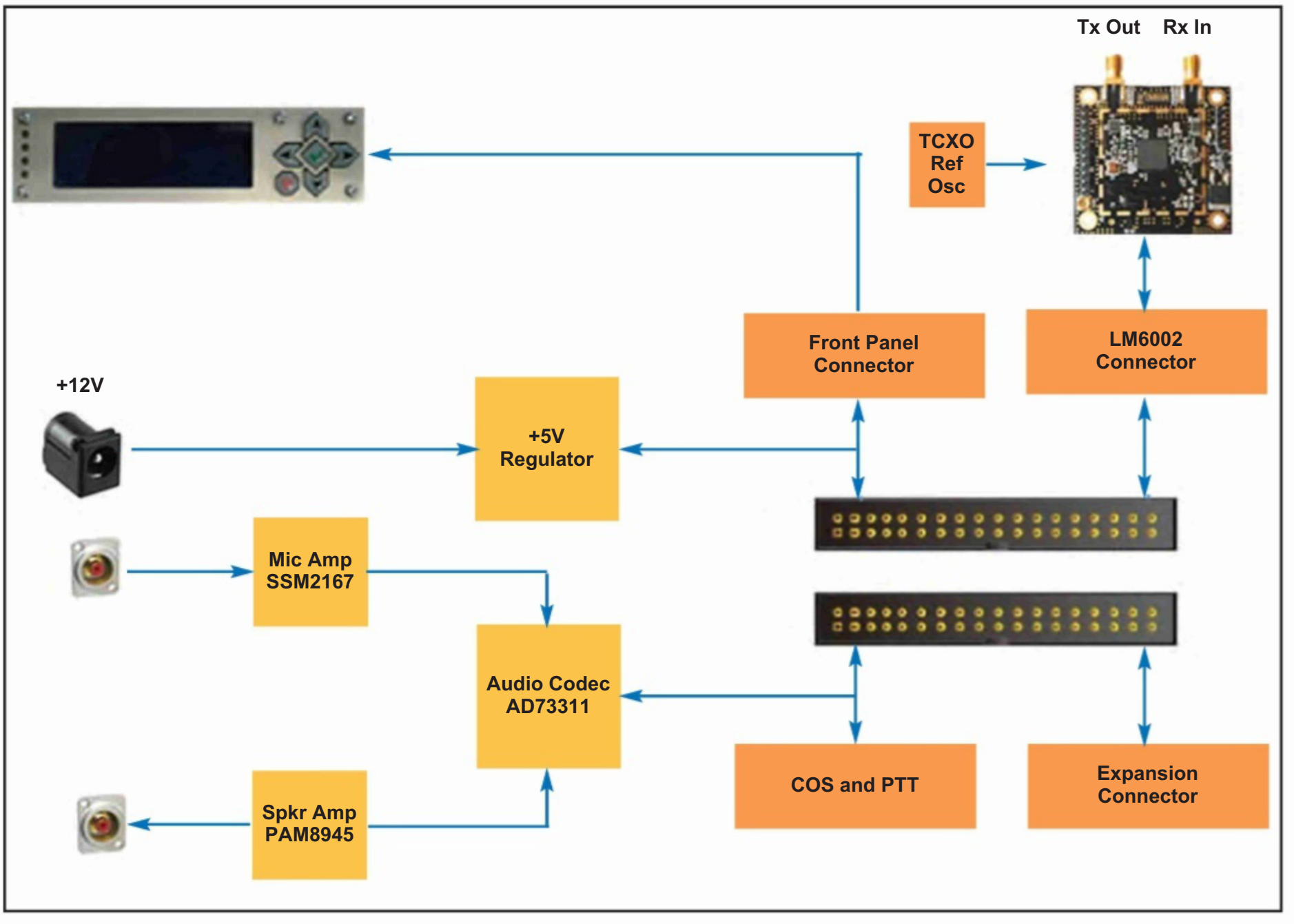


Figure 5. Mezzanine card block diagram

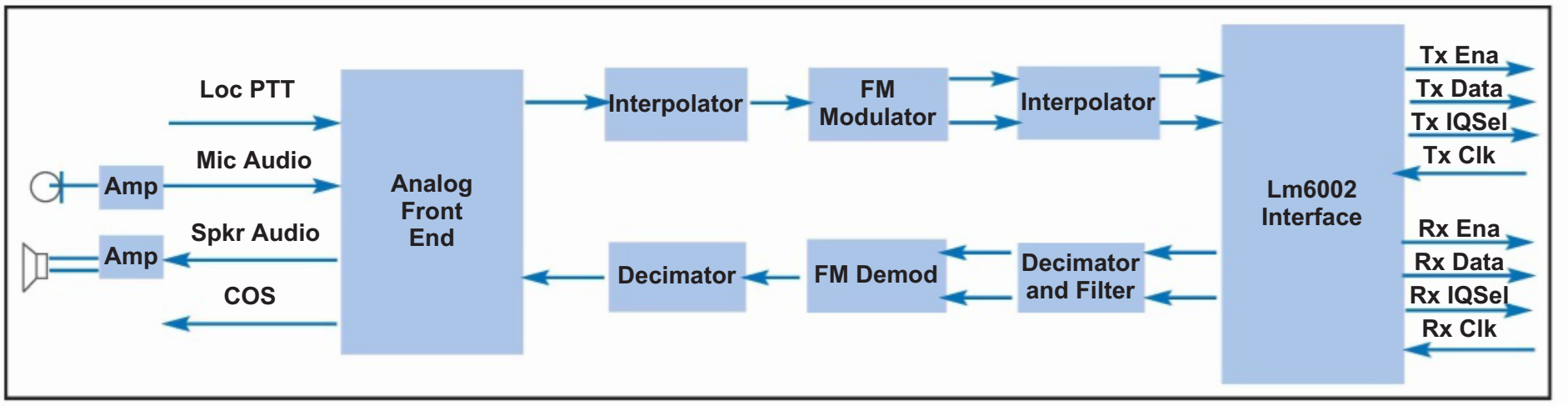


Figure 6. FM radio baseband processing

imum sample rate is twice that of the highest frequency that we want to represent. For good quality audio for this design, a rate of 25 kHz was chosen, which, by the Nyquist rule (theorem), will yield an audio and channel bandwidth of up to 12.5 kHz. We could also run the modulator at the Nyquist rate. However, more samples are more desirable which means that a higher sample rate is required, so it has to change between the front end and the modulator. This process is known as up-sampling or interpolation. The interpolator operates by taking in the audio samples at the 25-kHz rate and increasing that to 200 kHz by stuffing zeros in between the samples; seven zeros are inserted for every one input sample. Filtering is then done on the resulting stream to ensure that only the correct frequency components are present, and to also fill in the blanks. The signal processing structure to accomplish this task is called a *Cascaded-Integrator-Comb* (CIC), the topology of which is shown in *Figure 9*.

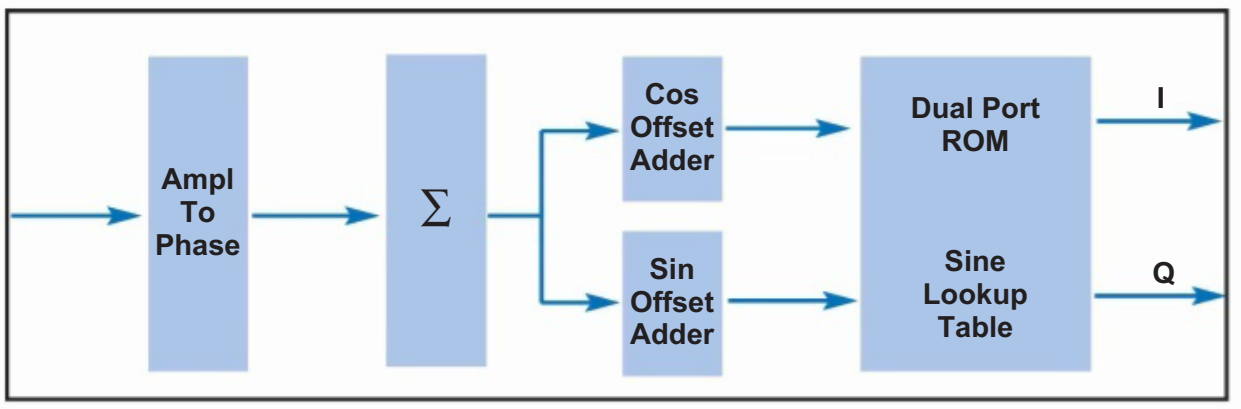


Figure 7. FM modulator

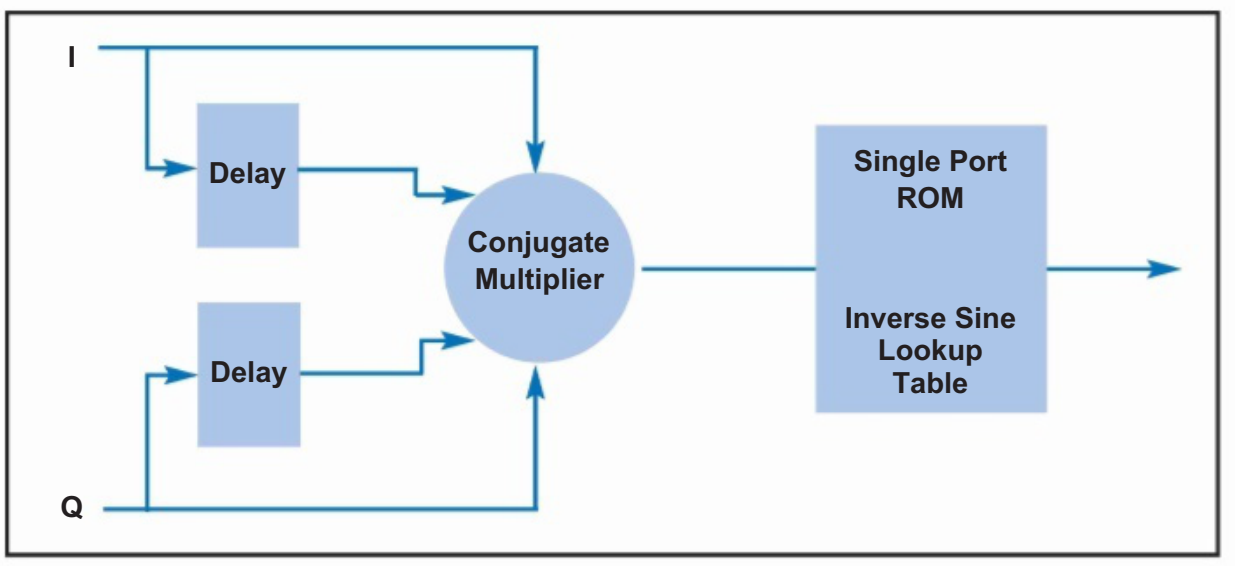


Figure 8. Phase demodulator

In this example, a third-order filter is shown, which consists of three "comb"' stages, a rate up-sampler, and three "integrator" stages. The attraction of this filter architecture is that it is has no coefficients or high-speed operations that need to be done, and it is made up of adders, sample delays, and counter logic which can be realized on the FPGA using macrofunctions which are generated by the development tools. Block diagrams of the integrator and comb stages can be seen in *Figure 10*.

In the receiver, we will need to do the opposite, which is called a decimator. Fortunately, we can use the same CIC structure but in reverse, with a down-sampler instead, as shown in *Figure 11*. The rate down-sampler simply "throws away" the unrequired samples from the higher rate to achieve the lower sample rate.

Between the RF module and the modulator/demodulator modules, we will need to make further sample rate changes. In the LM6002 on the Myriad board, the converters have a bandwidth of 1.5 MHz, which, by the Nyquist rule, requires that we sample at a minimum of 3 MHz. I prefer to operate at a higher rate, so 6 MHz was chosen instead. The post-modulator interpolation and pre-demodulator decimation then become a factor of 30, which can be implemented in two CIC stages, the first of 5 and second of 6. This rate is still rela-

Figure 9. Topology of a CIC filter

tively slow for an FPGA that can run upwards of 200 MHz.

Selectivity Filter

As the demodulator is running at 200 kHz, then by the same set of rules we can find components up to 100 kHz from adjacent channels. If the desired channel spacing is ±15 kHz, then additional filtering before the demodulator is required to narrow it. For this, we employ a *Finite Impulse Response* (FIR) filter, a structure that is quite computationally intensive, but does not suffer from arithmetic problems such as overflow. The filter takes in several samples, known as taps, and multiplies each of them by a set of coefficients, then sums these together. There are four major components in the filter: A sample memory, a coefficient memory, a multiplier/accumulator and some control logic, as can be seen in *Figure 12*.

For storing the samples, we use a dual-port RAM as a circular buffer. The read and write ports operate independently of each other, which enables the write logic to place a sample into the RAM, and the read logic to take out as many as are required for the filter operation. The coefficients are stored in a separate ROM, which has its own read logic. The input and its corresponding coefficient are then sent to the multiplier/accumulator. After all these are completed, the resulting sum is now the output. The filter needs to perform all its calculations for each input sample, so it must operate at a higher frequency. The clock rate for a 100-tap filter must be at least 20 MHz, which is still relatively slow for an FPGA. At higher rates we can duplicate the RAM and ROM structures and perform some operations in parallel.

Adding Linking

A desirable feature these days is to be able to generate *Pulse Code Modulation* (PCM) audio packets, which is a common telephony encoding method, for linking and connection to Allstar and Asterisk systems, or other Voice over IP (VOIP) applications. It has an 8-kHz sample rate and each sample consists of 8 bits, which have been compressed down from 12 bits. Two additional steps would be required to support it, the first would be to up- or down-sample, the second to compress or decompress 12 bits to 8 and vice versa. A decimation/interpolation of 25 using CIC filters again would solve the sample rate problem, and the PCM conversion can be accomplished with another lookup table. Data can be exchanged between the ARM processor and FPGA fabric using the

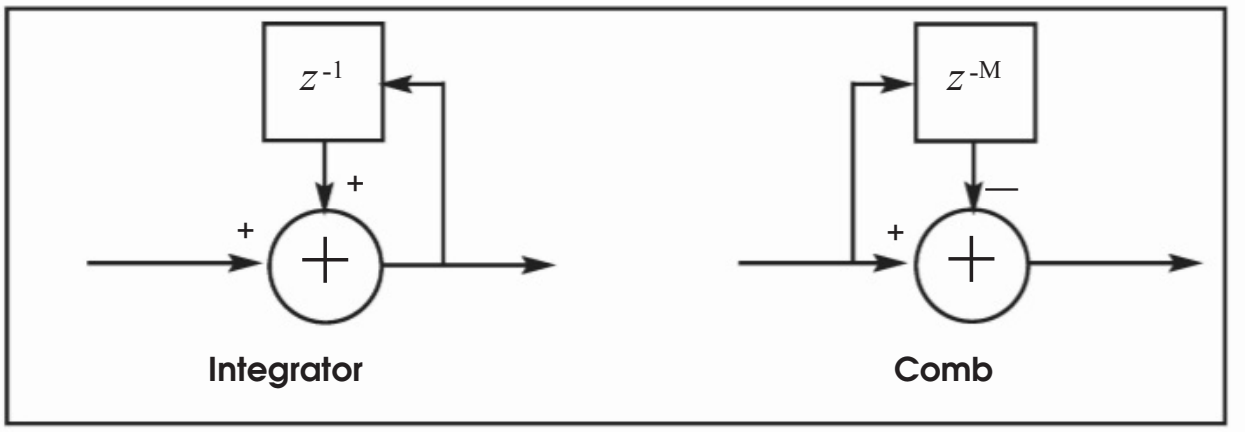
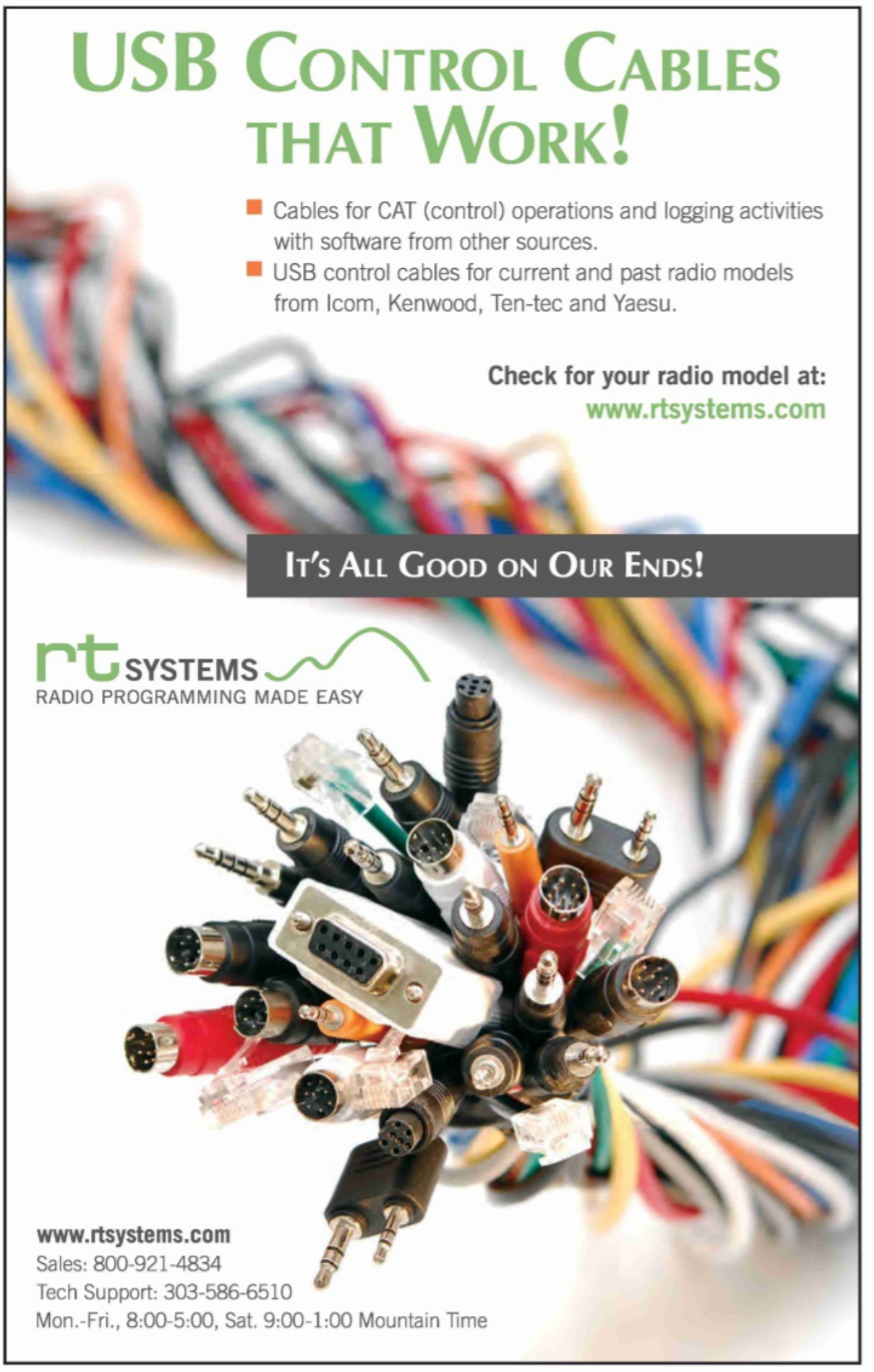


Figure 10. Comb and integrator structures


USB CONTROL CABLES THAT WORK!
Cables for CAT (control) operations and logging activities with software from other sources.
USB control cables for current and past radio models from Icom, Kenwood, Ten-tec and Yaesu.
Check for your radio model at:
www.rtsystems.com
IT'S ALL GOOD ON OUR ENDS!
rt SYSTEMS
RADIO PROGRAMMING MADE EASY
www.rtsystems.com
Sales: 800-921-4834
Tech Support: 303-586-6510
Mon.-Fri., 8:00-5:00, Sat. 9:00-1:00 Mountain Time

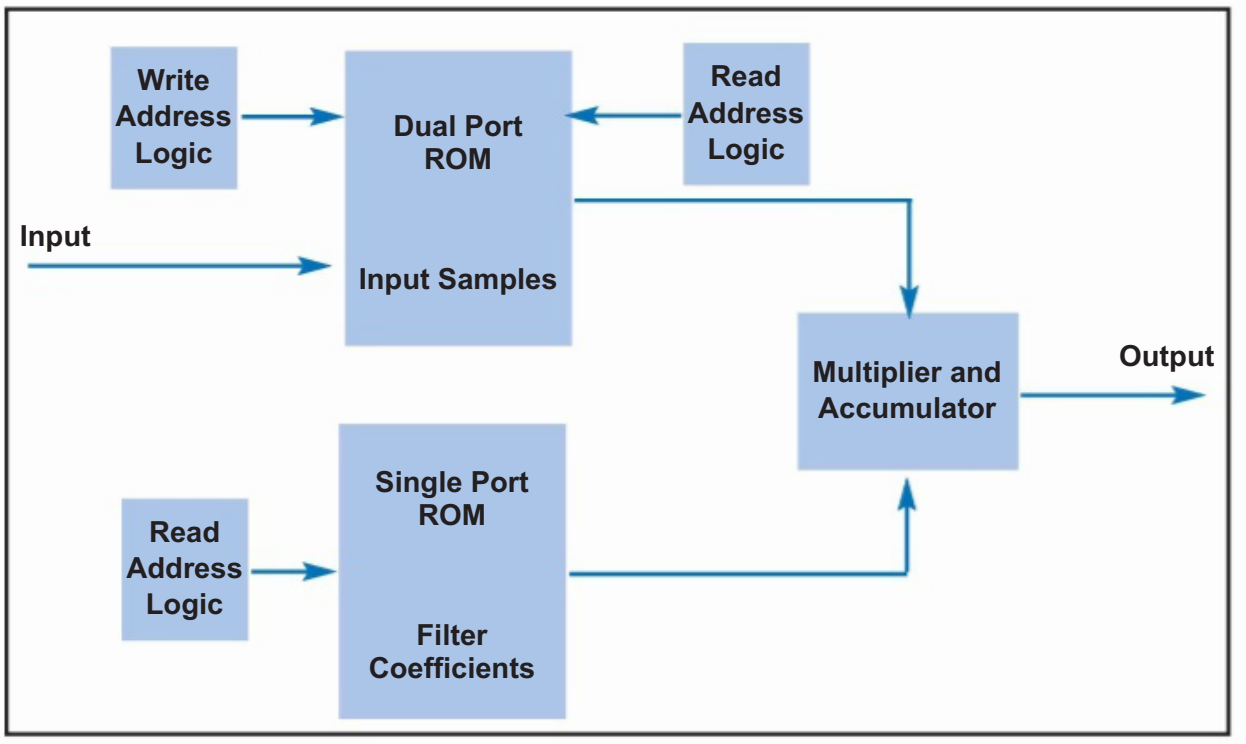


Figure 12. FIR filter structure

high-speed DMA channel, so the processor can assume the task of sending and receiving the ethernet packets and can make use of the full TCP/IP stack which is available under Linux.

Programming the FPGA

The design methodology for ASICS has evolved over the last 20 years to now make them more accessible to the average amateur. When I first started building them back in the late 1980s, we used schematic capture and familiar gate topologies like Boolean AND, OR, and flip-flops. Since then, two advances have arrived. The first was the invention of *Hardware Description Languages* (HDLs), such as VHDL (an HDL to describe Very Large Scale Integration devices, known as VLSI), and Verilog. The former has a syntax like that of the PASCAL language, the latter more like the familiar C programming language. The second was the technique of Synthesis, which takes the HDL code and generates gate level logic. All the components mentioned in this article were coded in the Verilog language and placed and routed on the FPGA using the tools available from the manufacturer. Surprisingly, it took only a small amount of the available resources: 10% of the logic, 3% of the memory and 16% of the DSP (multiplier) blocks.

Conclusion

The combination of an FPGA and RF module is a viable platform for developing inexpensive radio systems, using well-known signal processing structures that can be realized in a high-level language and implemented with freely-available tools. The FPGA has many resources that can enable more complex modulation schemes, and the hard processor can run amateur software such as Allstar, or mesh networking software such as Hamnet or an AREDN node. I wrote this article after developing a 900-MHz FM repeater as a simple example and do have plans to implement further modes in the future. What they will be is now only limited by the imagination!

Notes:

1. More information about the Terasic DE10-Nano can be found at <https://tinyurl.com/lcoyhz2>.
2. For more information on the Myriad Board, visit <https://tinyurl.com/qh7gbur>.

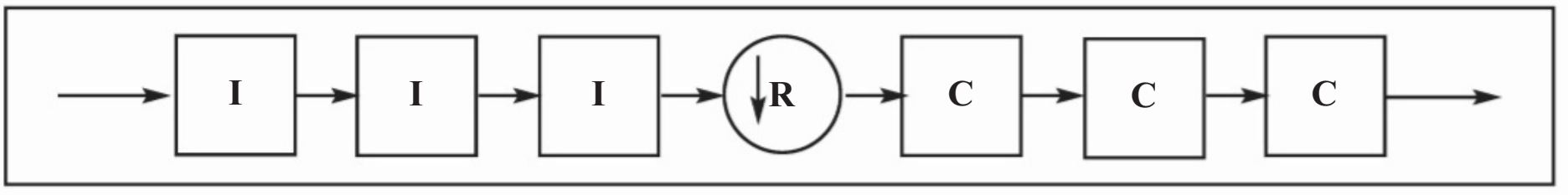


Figure 11. CIC decimator

what's new

SOTABEAMS Gutter Clips for Stealth Antennas

Many radio amateurs face restrictions from homeowners' associations when they want to install HF antennas at home. Often the best solution is a wire antenna installed around the rain gutter of a house. SOTABEAMS now stocks hooks that can be used to clip a wire around guttering. Made from translucent white plastic, the clips simply attach over the edge of the gutter.

SOTABeam's rain gutter clips are available now, come in packages of 36, and have a suggested retail price of $1.97 (For Europeans) $1.64 for everyone else. For more information, contact SOTABEAMS, Unit 1 The Green, Fountain Street, Macclesfield, SK10 1JN, UK. Phone: +44 (0) 1625 501643. Website: <https://tinyurl.com/yce4mfj5>.